Machine Learning Engineer Nanodegree
Capstone Report: Bertelsmann/Arvato Project

1. Introduction
In this project, we will study marketing and demographic data related to the German company Bertelsmann.
The data is mainly composed of 4 csv files representing:
- Demographic data of the general population of Germany
- Customers of the company
- Train dataset with users responding positively to a marketing campaign
- Test dataset to predict the outcome of a future campaign

The objective of this project is to help Arvato predict the response of a resident to their campaign in the future. And to do so, we will follow all the steps to tackle a machine learning project from start to finish (from data analysis to model building, optimization, and, in our case, submission to a Kaggle competition). A lot of testing/trial and error has been done to achieve the final code and results in the Jupyter notebook and will be discussed in the section Discussion, along with what did not work and what other steps could be taken further in the project.
In this report, we will start by discussing the environment setup, which includes the installed libraries. To gain a more profound knowledge of the data and spot any potential problems or trends, we will then do exploratory data analysis (EDA) on the datasets that have been provided. We will also create EDA reports to enumerate the features of the datasets.
We will use an automated library to visualize the training dataset in many different ways to get more information than the first EDA reports.
Next, we do a simple cleaning and transformation of the data, feature engineering, and data transformation to prepare it for the segmentation and modeling phases.
Finally, we will discuss the modeling strategy, including the assessment metric picked, the AUR (area under the curve) for the ROC curve (receiver operating characteristic), and the models selected and optimized.
The main goal of this report, besides showing an entire pipeline of ML projects, is to present the results, findings, shortcomings, and solutions proposed.

2. Environment Setup
A README.md file is included that explains the development environment used, i.e., the Python and Conda versions in this case, plus all the other libraries not included by default in Python. The simple way to install them is using pip install followed by their names. A requiremet.txt file, usually found with Python projects, is not included. If Conda (Miniconda) and Python versions are respected, there is little chance of reproducibility conflicts happening on a different architecture.
Help directories are also created in this step to host our EDA, visualization, and submission result files.

3. Exploratory Data Analysis (EDA)

After the environment is setup and all required libraries are loaded, the first and most important step in any machine learning project dealing with a dataset is to explore this dataset, understand it, and find any interesting characteristics or pitfalls that could negatively impact the full process afterward, from modeling to deployment or retraining, etc. To enhance this step, I opted to automate it using the library pandas-profiling, which extends the known.describe() function of dataframes in pandas into a more comprehensive analysis that includes almost all desired statistics and plots from an initial EDA.

The reports are exported into HTML file format. As it was performed on all four main datasets of customers, Azdias, train, and test, which are relatively big datasets for doing the full analysis by pandas profiling, it should be done using a good machine. The Jupyter might crash (especially if calculating the correlations), so a python file (eda.py) for that part alone in the project for execution on a VPS.

Here are some of the findings for two datasets: (Azdias and train)

- Udacity_AZDIAS_052018

Dataset statistics

Number of variables	366
Number of observations	891,221
Missing cells	33,492,923
Missing cells (%)	10.3%
Total size in memory	2.4 GiB
Average record size in memory	2.9 KiB

Variable types

Numeric	360
Categorical	4
Unsupported	2

400 warnings of the features include high cardinality, missing values, and zeroes.

Overview | Warnings 400 | Reproduction

Warnings

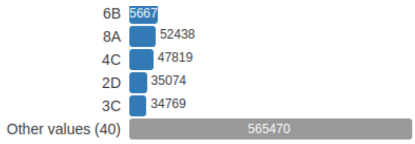
EINGEFUEGT_AM has a high cardinality: 5162 distinct values	High cardinality
AKT_DAT_KL has 73499 (8.2%) missing values	Missing
ALTER_HH has 73499 (8.2%) missing values	Missing
ALTER_KIND1 has 810163 (90.9%) missing values	Missing
ALTER_KIND2 has 861722 (96.7%) missing values	Missing
ALTER_KIND3 has 885051 (99.3%) missing values	Missing
ALTER_KIND4 has 890016 (99.9%) missing values	Missing
ALTERSKATEGORIE_FEIN has 262947 (29.5%) missing values	Missing
ANZ_HAUSHALTE_AKTIV has 93148 (10.5%) missing values	Missing
ANZ_HH_TITEL has 97008 (10.9%) missing values	Missing
SOHO_KZ has 810834 (91.0%) zeros	Zeros
TITEL_KZ has 815562 (91.5%) zeros	Zeros
UNGLEICHENN_FLAG has 744072 (83.5%) zeros	Zeros
VERDICHTUNGSRAUM has 368782 (41.4%) zeros	Zeros
VHA has 665547 (74.7%) zeros	Zeros
VHN has 36868 (4.1%) zeros	Zeros
W_KEIT_KIND_HH has 40386 (4.5%) zeros	Zeros

A random variable is chosen with the following analysis:

CAMEO_DEU_2015
Categorical

MISSING

Distinct	45
Distinct (%)	< 0.1%
Missing	98979
Missing (%)	11.1%
Memory size	6.8 MiB



Toggle details

Overview | Categories

Common Values

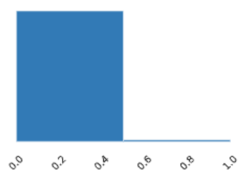
Value	Count	Frequency (%)
6B	56672	6.4%
8A	52438	5.9%
4C	47819	5.4%
2D	35074	3.9%
3C	34769	3.9%
7A	34399	3.9%
3D	34307	3.8%
8B	33434	3.8%
4A	33155	3.7%
8C	30993	3.5%
Other values (35)	399182	44.8%
(Missing)	98979	11.1%

- Udacity_MAILOUT_052018_TRAIN

RESPONSE
Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Distinct	2	Minimum	0
Distinct (%)	< 0.1%	Maximum	1
Missing	0	Zeros	42430
Missing (%)	0.0%	Zeros (%)	98.8%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	0.01238303617	Memory size	335.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Value	Count	Frequency (%)
0	42430	98.8%
1	532	1.2%

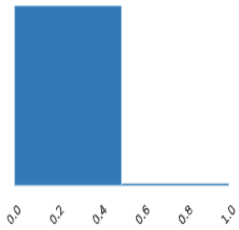
The analysis for the RESPONSE variable in the training dataset as an example:

RESPONSE

Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Distinct	2	Minimum	0
Distinct (%)	< 0.1%	Maximum	1
Missing	0	Zeros	42430
Missing (%)	0.0%	Zeros (%)	98.8%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	0.01238303617	Memory size	335.8 KiB



Toggle details

Statistics Histogram Common values Extreme values

Value	Count	Frequency (%)
0	42430	98.8%
1	532	1.2%

Most of the giving data has negatively responded to the campaign.

4. Visualization of Data

I have also used an automated library for visualization that also suggests the preprocessing that one might apply to the data at the beginning. I used it exclusively on the train dataset (which includes the variable target Response).

Extract of the suggestions (full list in notebook cell output):

```
Data cleaning improvement suggestions. Complete them before proceeding to ML modeling.
```

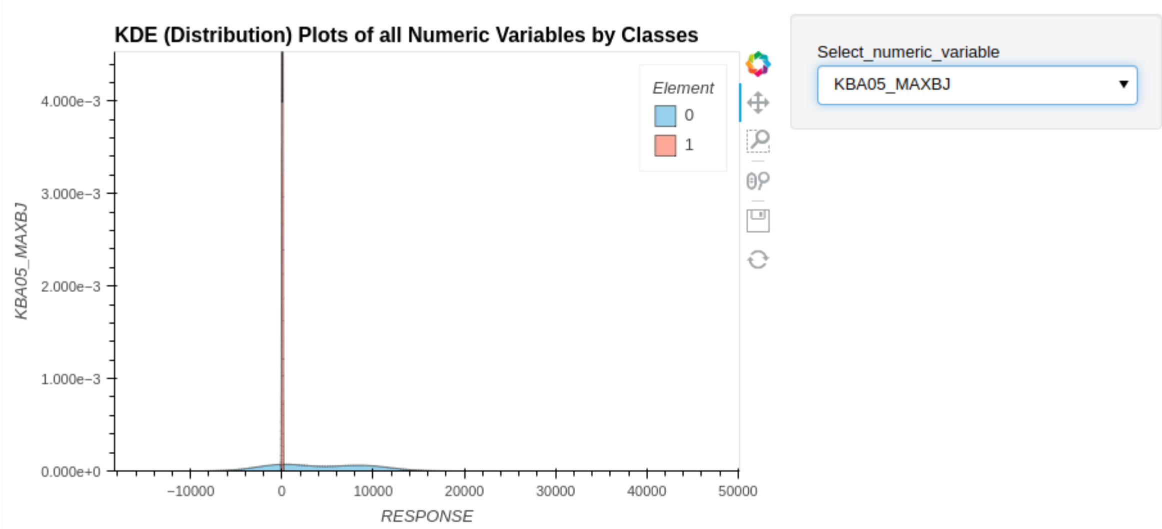
	Nuniques	dtype	Nulls	Nullpercent	NuniquePercent	Value counts Min	Data cleaning improvement suggestions
EXTSEL992	56	float64	15948	37.121177	0.130348	0	fill missing
VERDICHTUNGSRAUM	46	float64	7955	18.516363	0.107071	0	fill missing, skewed: cap or drop outliers
EINGEZOGENAM_HH_JAHR	33	float64	6969	16.221312	0.076812	0	fill missing, skewed: cap or drop outliers
MIN_GEBAEUDEJAHR	31	float64	7777	18.102044	0.072157	0	fill missing, skewed: cap or drop outliers
ALTERSKATEGORIE_FEIN	25	float64	8155	18.981891	0.058191	0	fill missing
ALTER_HH	20	float64	6969	16.221312	0.046553	0	fill missing
ANZ_HH_TITEL	15	float64	8246	19.193706	0.034915	0	fill missing, highly skewed: drop outliers or do box-cox transform
GFK_URLAUBERTYP	12	float64	605	1.408221	0.027932	0	fill missing
AKT_DAT_KL	9	float64	6969	16.221312	0.020949	0	fill missing, skewed: cap or drop outliers
D19_BANKEN_ONLINE_QUOTE_12	8	float64	7584	17.652809	0.018621	0	fill missing, skewed: cap or drop outliers
INNENSTADT	8	float64	7799	18.153252	0.018621	0	fill missing
KBA05_ZUL3	7	float64	8648	20.129417	0.016293	0	fill missing
KBA05_VORB2	7	float64	8648	20.129417	0.016293	0	fill missing
D19_KONSUMTYP	7	float64	7584	17.652809	0.016293	0	fill missing, skewed: cap or drop outliers
D19_LOTTO	7	float64	7584	17.652809	0.016293	0	fill missing
KBA05_ALTER4	7	float64	8648	20.129417	0.016293	0	fill missing
KBA05_MOD4	7	float64	8648	20.129417	0.016293	0	fill missing
KBA05_HERST4	7	float64	8648	20.129417	0.016293	0	fill missing
BALLRAUM	7	float64	7799	18.153252	0.016293	0	fill missing
KBA05_HERST5	7	float64	8648	20.129417	0.016293	0	fill missing
ANZ_KINDER	7	float64	6969	16.221312	0.016293	0	fill missing, highly skewed: drop outliers or do box-cox transform
KBA05_SEG10	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA05_SEG3	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA05_SEG4	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA05_VORB0	6	float64	8648	20.129417	0.013966	0	fill missing
KBA05_VORB1	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA05_MOD3	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA05_ZUL1	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA13_CCM_0_1400	6	float64	7962	18.532657	0.013966	0	fill missing
KBA05_ZUL2	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA13_BJ_2008	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_BJ_2009	6	float64	7962	18.532657	0.013966	0	fill missing
KBA05_MAXHERST	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA13_CCM_1000	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_CCM_1200	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_CCM_1800	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_CCM_2500	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_CCM_2501	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_CCM_3000	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_KRSAQUOT	6	float64	7962	18.532657	0.013966	0	fill missing
KBA13_KRSHERST_AUDI_VW	6	float64	7962	18.532657	0.013966	0	fill missing
KBA05_MOD2	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
KBA05_MODTEMP	6	float64	7777	18.102044	0.013966	0	fill missing
GEBAEUDETYP	6	float64	7777	18.102044	0.013966	0	fill missing, skewed: cap or drop outliers
KBA05_CCM2	6	float64	8648	20.129417	0.013966	0	fill missing, skewed: cap or drop outliers
ARBEIT	6	float64	7951	18.507053	0.013966	0	fill missing
CJT_GESAMTTYP	6	float64	605	1.408221	0.013966	0	fill missing
D19_SOZIALES	6	float64	7584	17.652809	0.013966	0	fill missing
KBA05_MAXAH	6	float64	8648	20.129417	0.013966	0	fill missing
HH_EINKOMMEN_SCORE	6	float64	704	1.638657	0.013966	0	fill missing

The library generated 6 HTML files for the visualization :

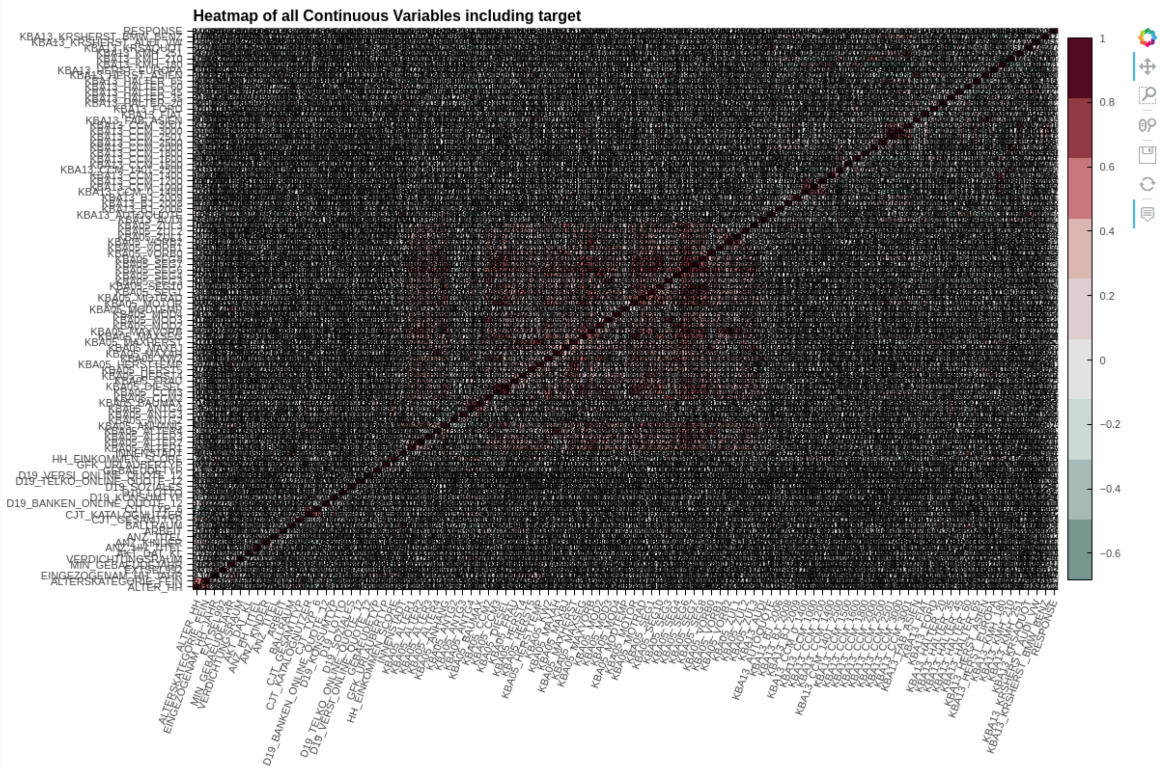
- Distplots_nums.html
- Heatmaps.html
- Kde_plots.html
- Pair_scatters.html
- Scatterplots.html
- Violinplots.html

Note: Pair_scatters.html is 3.4 GB and is not attached to the solution.

- Displot

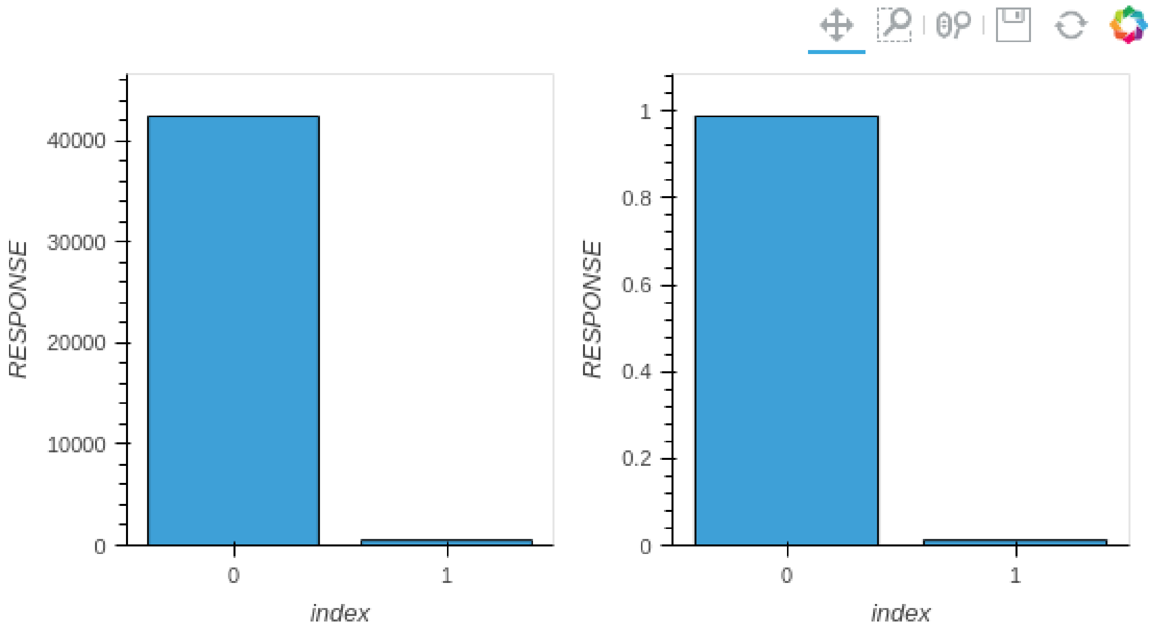


- Heatmap

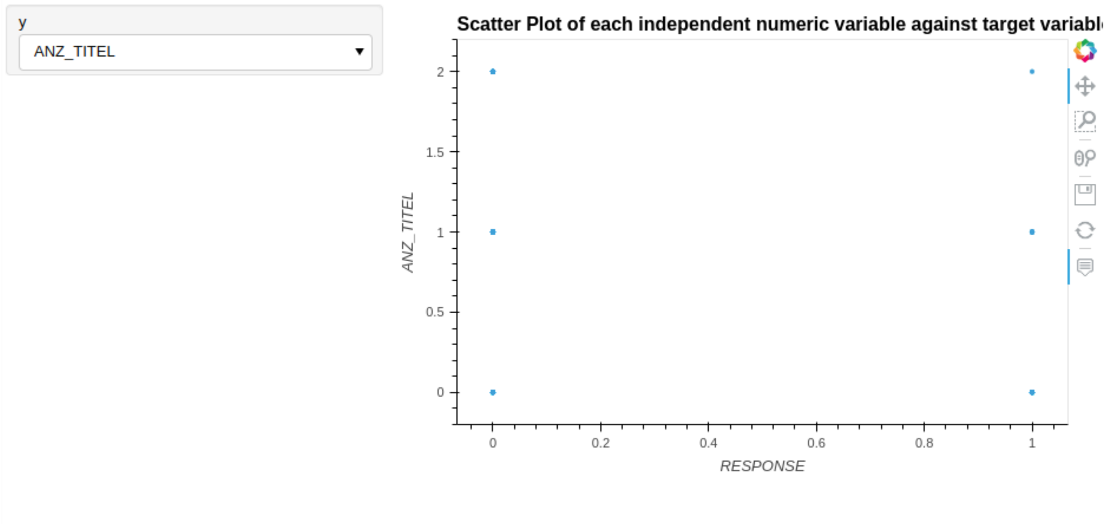


- KDE plots

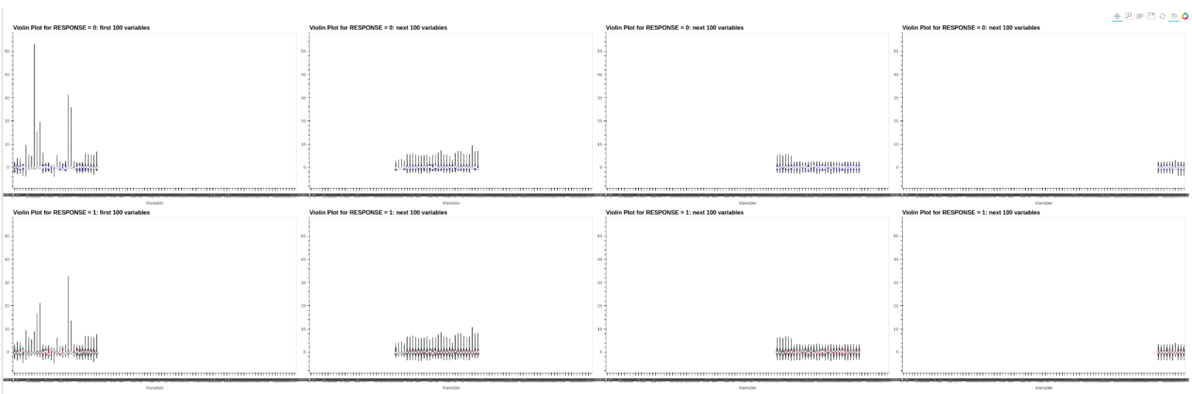
Histogram and KDE of Target = RESPONSE



- Scatter plots



- Violin plots



Note: As the process is automated, the types of variables are not all respected.

5. Cleaning of Data

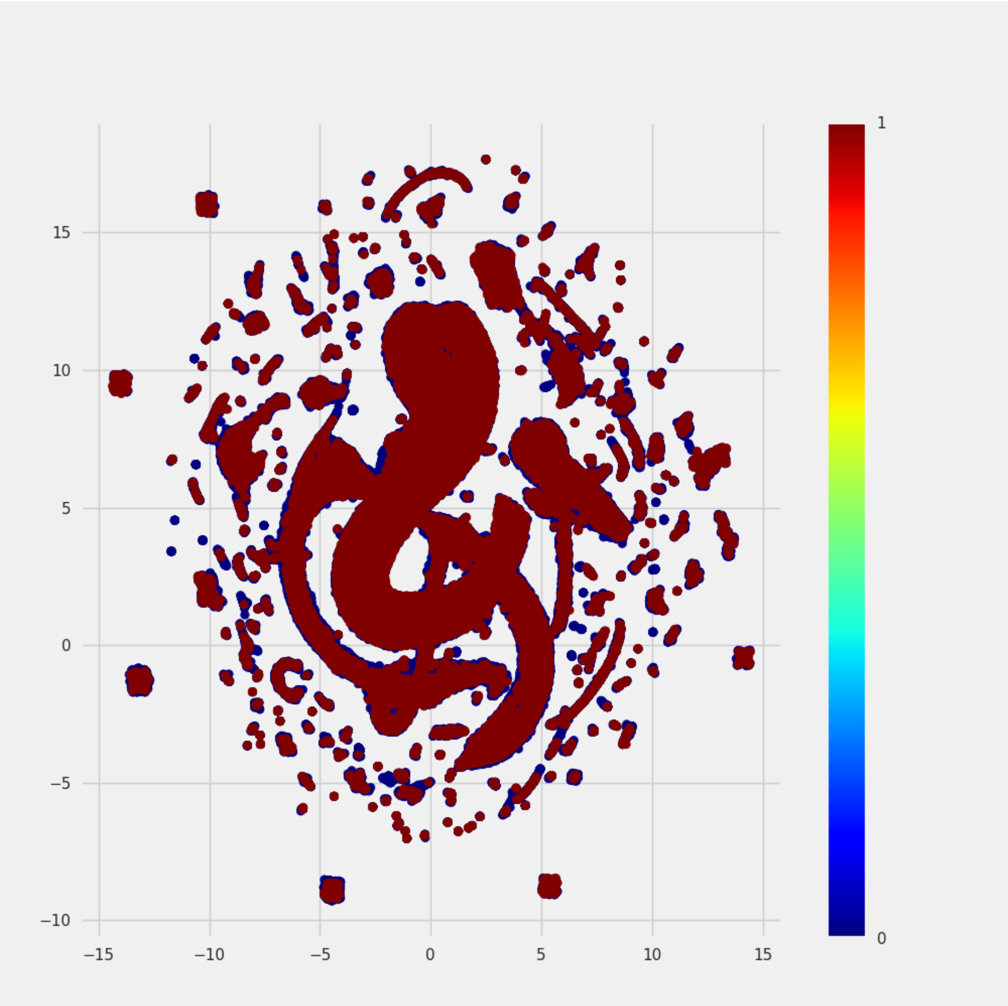
I chose to do a simple cleaning of the data to preserve as much signal as possible for modeling and leave it to optimization to choose the best model. Both the EDA and Viz reports show a lot of missing data. A function clean() is created to do all the data cleaning for the 4 CSV files, including dropping duplicate rows, dealing with missing values, and encoding a date time variable. We also use one-hot encoding for encoding all categorical variables in datasets.

After this step is done, I move on to the next one, which is segmentation.

6. Segmentation

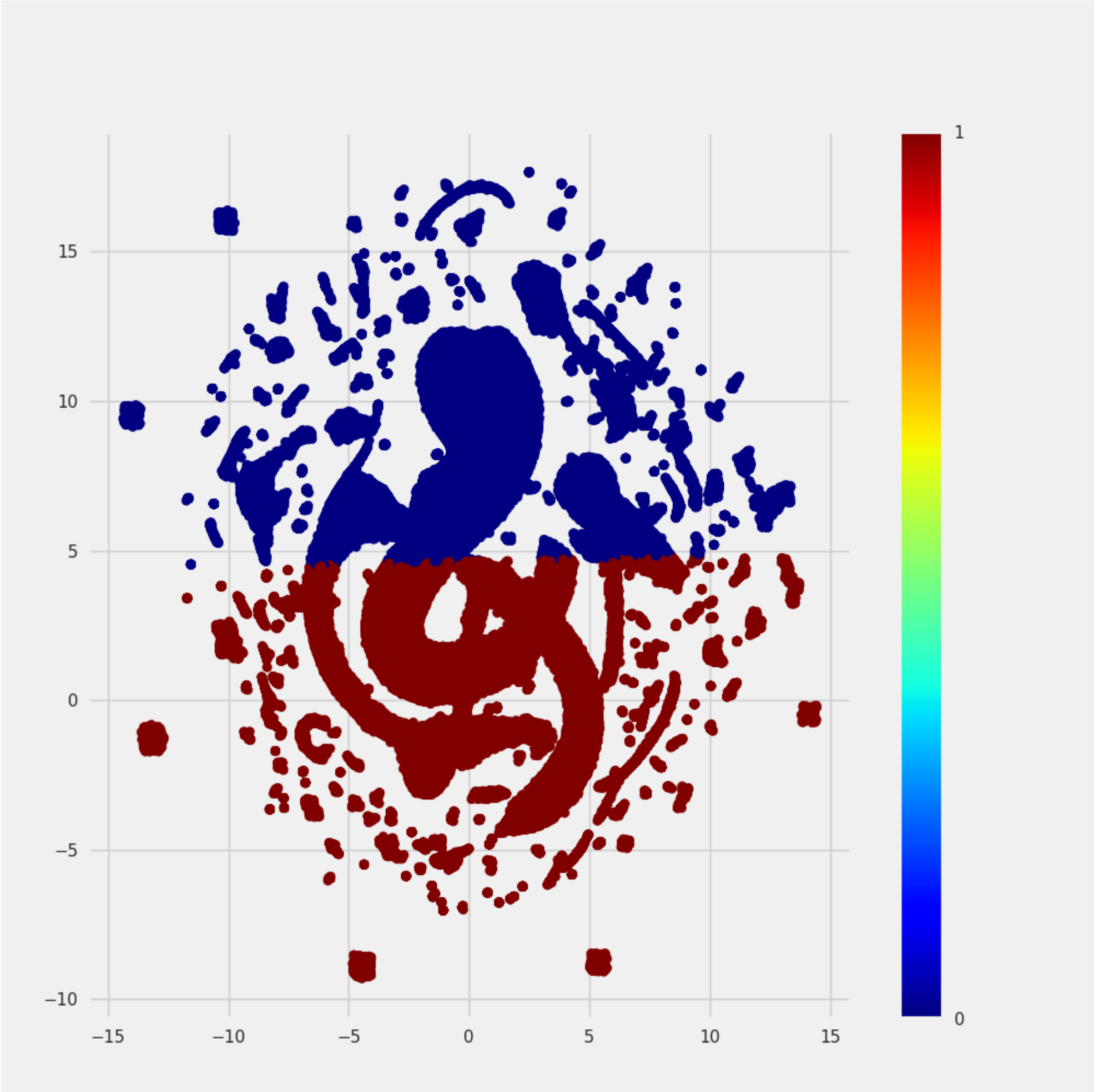
In this step, the goal is to decide which residents of Germany in the Azdia dataset are similar to the customer of the company in question. To do this, I concatenated both datasets into a single dataframe and then applied a dimensionality reduction to 2D space; I have done a lot of tests with TSNE from sklearn package and the one with Barnes-Hut, tsne_cuda but found UMAP is the best for this dataset and the one that is faster and does not crash.

This is the plot of the result.



Coloring is based on whether a person is a customer or not.

Then I perform K-means to segment all observations in the 2D projection space to 2 clusters:



Color is based on whether in cluster 0 or cluster 1

Because I have the index of the customers before concatenation, I have checked the intersection between each cluster and the existing known customers; I have decided that the intersection with the biggest number is the one more likely to be a future customer of the company, while the other cluster is less likely to do so.

More clustering finetuning can be done, of course, to enhance these results.

7. Modeling

Before modeling the data, it needs preprocessing after the initial cleaning I used the clean() function. That is why a step using standard scaler is applied before feeding it into any machine learning model as a form of regularization of the different feature ranges.

I chose as a benchmark model the logistic regression model from the sklearn library. Its cross-validation (CV) score was 0.64 (ROC_AUC).
I decided to go with boosting trees, so I chose Lightgbm as the main model for this classification problem to model the training dataset. To finetune it from the beginning, I use the library scikit-optimize which does hyperparameter optimization of a given hyperspace for the models using cross-validation. I also used an xgboost model to compare both (it was finetuned with the same method). However, I found that lightgbm gave the best results in the shortest training time, which is why I have constrained the number of iterations for xgboost and fold compared to lightgbm.
To make the solution more robust, I did a blend (ensembling) of both models. Usually, in literature, this technique always gives the best solution.
The CV score for LGBM is 0.77, and the one for XGB is 0.73. Indeed both models are better than vanilla logistic regression

8. Kaggle Submission

I have submitted both predictions of lgbm and xgb models and their ensembles to the Kaggle competition, and the results of the public and private scores for the three submissions are below.

Submission and Description	Private Score	Public Score	Selected
submission_ensemble.csv Complete (after deadline) · now	0.72422	0.77796	☐
submission_xgb.csv Complete (after deadline) · 1m ago	0.71051	0.76339	☐
submission_lgbm.csv Complete (after deadline) · 2m ago	0.75648	0.79937	☐

9. Conclusion

A lot of work can be done to enhance the results of this project, from more preprocessing, finetuning, model variations, clustering, etc. However, this project has designed a complete roadmap to tackle any machine learning project from step 1 to the last step of prediction while maintaining a result and business-driven orientation. That is why the value of both finding similar persons to the existing customers and predicting in advance a customer response to a marketing strategy is of utmost benefit and importance to the business side of the company.